Alkalmazott Matematikai Lapok **23** (2006), 313–333.

KRIPTOGRÁFIAI CÉLÚ VÉLETLENSZÁM GENERÁLÁS ÉS ELLENŐRZÉS

BORBÁS GERGELY, NEMETZ TIBOR, PAPP PÁL

1. Bevezetés

Üzenetek titkosításához egyértelműen dekódolható kódok egy halmazát, a kódhalmazt szokták megválasztani. Kerckhoff 19. századi munkássága óta ezt a halmazt ismertnek tételezik fel a rejtjelfejtés számára. Szimmetrikus kulcsú rendszerek esetében egyetlen titokban tartandó elem az, hogy a sok kód közül melyiket használjuk egy konkrét üzenet titkosítására. A felhasznált kód megadására annak egy azonosítóját használják fel. Ezt az azonosítót **az üzenet** (a kiválasztott kód) **kulcsának** nevezik. A kulcsot az üzenet továbbítója és címzettje az üzenetváltás előtt úgy egyeztetik egymással, hogy egy harmadik személy a kiválasztott kulcsról semmilyen információt ne tudjon szerezni.

Egy harmadik személy számára tehát a kulcs választása *maximális bizonytalanság*ot kell, hogy tartalmazzon. Ezért az aktuális **kulcsot véletlenszerűen kell kiválasztani** az összes lehetséges kulcs halmazából, a **kulcstérből**. Nem jelent semmilyen megszorítást, ha a kulcsteret egymás után következő, nem-negatív egész számok halmazának tekintjük. Ennek a véges halmaznak kell egy elemét véletlenül (véletlenszerűen) kiválasztani.

Véletlenszerűen választott számokat a mindennapi élet, a számítástechnika más területein is régóta alkalmaznak, utalunk Knuth (1987) számítógépes bibliájára. Vannak olyan területek is, ahol véletlenszámok nélkül az alkalmazás hatékonyságáról nem is lehetne beszélni. A véletlen kulcsválasztás fontossá válása új lendületet adott a véletlenszám-generálásnak, új utakat nyitott meg.

A kriptográfiában véges sok értéket felvevő egyenletes eloszlású véletlenszámokkal dolgoznak. A felhasználandó véletlenszámok legalapvetőbb tulajdonsága a kiismerhetetlenség. Ez azt jelenti, hogy hiába ismerjük meg pontosan a korábban

már használt véletlenszámokat, a pillanatnyilag előállított (előállítandó) véletlenszámokról semmit sem tudunk, azok maximális bizonytalanságot tartalmaznak.

Heurisztikusan ezt úgy fogalmazhatjuk meg, hogy a változók a lehetséges értékek bármelyikét felvehetik, mégpedig egyenlő valószínűséggel (ezért nevezzük őket **egyenletes**nek), és teljesen függetlenül attól, hogy korábban milyen értékek jelentek meg (ezért hívjuk őket **független**nek). Etalonként gondolhatunk egy szabályos érme egymás utáni dobálása során nyert fej-írás sorozatokra (amelyekből a szokásos átírás után keletkezhetnek a **0–1 értékű bináris sorozatok**), vagy egy szabályos kocka dobásával nyert $1, 2, \ldots, 6$ értékeket felvevő sorozatokra. Ezek a sorozatok egy fizikai jelenség egymás utáni végrehajtása során adódnak, innen származik az elnevezésük is: **fizikailag generált** vagy **valódi véletlenszámok**. A fogalmat a következő definíció teszi matematikailag precízzé:

Definíció. Legyen $X = \{0, 1, \ldots, k-1\}$ az $X_1, X_2, \ldots, X_n$ valószínűségi változók közös értékkészlete. A változókat *teljesen független* (röviden független) *egyenletes eloszlású valószínűségi változó sorozatnak*, röviden **véletlennek** nevezzük, ha a lehetséges értékek bármilyen $x_1, x_2, \ldots, x_n$ választása mellett fennáll a

$$\mathbf{Prob}\{X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n\} = (1/k)^n$$

azonosság.

A kulcsválasztás feladatának megoldásához viszonylag kisszámú véletlenszám elegendő volt. Az elméletileg is fejthetetlen „one time pad" rejtjelző algoritmus (lásd később) azonban ugyanolyan hosszú véletlenbetű sorozatot igényelt, mint maga a nyílt szöveg (a titkosítandó üzenet), tehát általában nagyon hosszút. Ezt még ráadásul rendkívül biztonságosan szállítani és őrizni is kellett. Így természetes, hogy a felhasználók „takarékosan" igyekeztek felhasználni a véletlen betűket. Ezért az egy üzenet titkosításához már felhasznált véletlen sorozatot kisebb-nagyobb eltolásokkal ismételten felhasználták. Az ismételt kulcsfelhasználás azonban betűnkénti helyettesítés esetén már az I. Világháború idején is felismerhető volt a koincidencia teszt segítségével, lásd Friedman (1920). Kulcsismétléses táviratok fejtésére viszont még korábbról létezett megoldás, a „kerchkoffozás", lásd Kerchkoff (1883).

Az ismételt felhasználás a Szovjetunió esetében súlyos következményekkel járt. Már a II. világháború alatt az USA egy erre a célra betanított csapata folyamatosan fejtette a szövetséges Szovjetunió legtitkosabb üzeneteit. Ez a tény ugyan csak a 90-es években vált publikussá, de szakberkekben ismertté vált a veszély. Így az ismételt felhasználást más takarékos módszerrel igyekeztek kiváltani. Ennek a módszernek a lényege az volt, hogy a valódi véletlen sorozatokat olyan algoritmikusan generálható sorozatokkal helyettesítettek, amelyek „úgy viselkedtek, mintha véletlenek lennének." Ilyen sorozatok iránti igény találkozott a lényegében ugyanebben az időben keletkezett „polgári" igényekkel.

A számítógépek hőskorában rendkívül fontos számítástechnikai követelmény volt, hogy véletlenszerű számsorozatokat ismételten elő lehessen állítani, reprodukálni. A véletlen számsorozatok szerepében használt reprodukálható sorozatoktól

azt várták el, hogy *statisztikailag ugyanúgy viselkedjenek, mintha valódi véletlen sorozatok lennének* Ezért helyettük determinisztikus számsorozatot állítottak elő. Történelmileg először az atombomba gyártásánál merült fel nagyon nagy mennyiségű, reprodukálható véletlenszám előállításának az igénye. A magyar származású Neumann János 1944-ben azt javasolta, hogy erre a célra álvéletlen számokat használjanak.

Az algoritmikusan előállítható, véletlenszerűen viselkedő **számsorozatokat *álvéletlen* (pszeudovéletlen) sorozatoknak** nevezik. A számítógépek jelenlegi gyakorlata ilyen véletlenszámokkal dolgozik, csak imitálja a valódi véletlenszámok generálását, lásd Knuth (1987).

Definíció. Az $y_1, y_2, \ldots, y_n$ számsorozatot determinisztikusnak nevezzük, ha a sorozat bármely tagja meghatározható a megelőzők determinisztikus függvényeként, tehát létezik olyan f_n függvénysorozat, melyre

$$y_n = f_n(y_1, y_2, \ldots, y_{n-1}).$$

A véletlenszámok gyakorlatában (és irodalmában) nem szokták matematikai precízséggel definiálni az álvéletlen fogalmát. Mi ezt később, a statisztikai tesztek kapcsán fogjuk megtenni.

A kriptográfiai alkalmazásokban a tökéletes titkosságot csak **fizikai forrás alapján** generált véletlenszámok lehet biztosítani. Ennek megfelelően a nyilvános kulcsú kriptográfia terjedésével párhuzamosan a piacon az utóbbi évtizedekben számos fizikai generátor jelent meg. Ugyanakkor megnőtt az igény megbízható kész sorozatok iránt is. Egy ilyen helyzet mindig kompromisszumot eredményez, és ez most is így van. A hőskori táblázatok mintájára a fizikai forrás alapján generált sorozatokat **CD-ROM-ra** (DVD-re, vagy bármilyen más, nagy kapacitású adathordozóra) **írva** is be lehet szerezni, miközben *minden CD-ROM egy-egy új táblázatot jeleníthet meg.* Az ilyen típusú felhasználások vezérfonala az, hogy nagy mennyiségű valódi véletlenszámot biztonságos körülmények között generálnak, azokkal egy CD-ROM-ot teleírva tárolják őket, majd „jól összekutyult", *egyszeri kivételi algoritmussal* akkora részt vesznek ki belőle, amekkorára éppen szükség van, lásd Nemetz–Papp (1998).

Ahogyan arra már fentebb utaltunk, meg kell fogalmazni, mit jelent a kriptográfia számára az a kifejezés, hogy *véletlenszerűen viselkedik* egy számsorozat. Ehhez összegyűjtötték azokat a lényeges tulajdonságokat, amelyekkel a valódi véletlen sorozatok rendelkeznek. Ezeket a tulajdonságokat csoportosították, és meglétük ellenőrzésére a sorozatoktól függő függvényeket, „statisztikákat" határoztak meg. A statisztikák értékeiről kiszámították, milyen eloszlás szerint viselkednek, ha a sorozat valóban véletlenszerű. Ezt használják fel a véletlenszerűség ellenőrzésére, lásd **„Statisztikai próbák a kriptográfiai alkalmazásoknál használt véletlen- és pszeudo-véletlen szám generátorokra"**, NIST: Special Publication 800-22 (2001).

Definíció. Egy determinisztikus sorozatot relatív pszeudo véletlen sorozatnak tekintünk, ha kielégíti egy adott, a véletlen sorozatokra definiált statisztikai próbák rendszerét a véletlen sorozatoktól megkívánt szinten.

Meg kívánjuk jegyezni, hogy a pszeudo-véletlen fogalom valamennyi értelmes felhasználásban kimondva-kimondatlanul szerepel a háttérben egy ilyen statisztikai teszt-csomag.

Tanulmányunk következő fejezeteiben továbbra is a kriptográfiai igényekhez igazodva ismertetjük a véletlenszámok előállításának és ellenőrzésének módjait. Mindezt kiegészítjük **két új véletlenszám generátorral**, és egy olyan **programcsomaggal**, amely az USA szabványoknak megfelelően a felhasználásra számot tartó **sorozatok véletlenszerűségét ellenőrizni tudja.**

Mindhárom megoldás *jogtiszta alkalmazást* tesz lehetővé, megszabadítva az alkalmazót az Internetről letölthető, vagy egyéb úton beszerezhető megoldásokban esetleges rejlő tulajdonjogi buktatóktól.

2. Véletlenszámok generálása: helyzetkép

Felidézzük, hogy a kriptográfiában véletlenszámok sorozata alatt heurisztikusan a következőt szokás érteni. Valamilyen k egész szám mellett tekintsük a $\{0, 1, \ldots, k-1\}$ értékkészletet. Tekintsük ebbe a halmazba tartozó számok olyan sorozatát, amelyeknek mindegyike lényegében ugyanannyiszor fordul elő a sorozatban, mégpedig bármelyik helyen bármelyik szám **egyenlő eséllyel** fordul elő (*egyenletesség*), **függetlenül attól**, hogy mi volt előtte, vagy mik jönnek utána (függetlenség). Az értékkészlet k számosságának szokásos megválasztása $k = 2$, amikor **bináris véletlenszámok**ról beszélünk. Ezek központi jelentőségét az információelméletben figyelhetjük meg. Ugyancsak gyakori a $k = 10$ választás, amikor **decimális véletlenszámok**ról beszélünk. A $\{0, 1, \ldots, k-1\}$ értékkészlet helyett egy nyelv ábécé-jét is lehet értékkészletnek tekinteni. Ekkor is véletlenszámokról, vagy véletlenbetűkről beszélnek.

Véletlenszámok manuális előállításához kockát, pénzérmét, kártyát, sorsolást szoktak használni. Ez azonban egy rendkívül lassú folyamat. Ezért keresnek más lehetőségeket is. Századunk első felében a statisztikai következtetésekhez a véletlenszámokat manuálisan állították elő és táblázatolták. Az első „nagy" táblázatot az angol Tippet készítette 1927-ben: 40 ezer véletlen számjegyet állított elő népszámlálási adatokból. 1939-ben 100 000 véletlen számjegyet állítottak elő egy erre a célra épített célgéppel. 1955-ben a RAND CORPORATION 1 millió véletlen számjegyet publikált.

2.1. Véletlenszámok fizikai generálása

Kriptográfiai kulcsok generálásakor a pszeudó-véletlenszám generátorok előnyei rendkívüli hátrányokká válnak. Ilyenkor különösen fontos, hogy

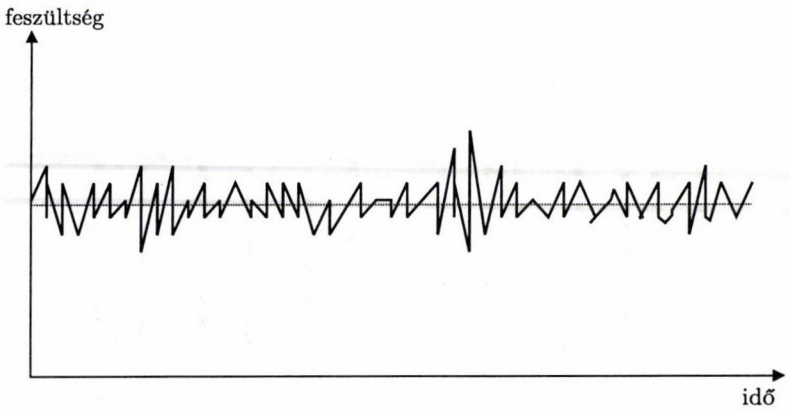
- az elkészült sorozat ne legyen reprodukálható,

- „információtartalma" megegyezzen a méretével.

Fentiek miatt pusztán fizikai véletlenszám generátorokat használnak a rejtjelkulcsok készítésére. ***A fizikai véletlenszám generátorok alapjául véletlen jeleket adó természeti jelenség szolgál.***

A gyakorlatban legtöbbször valamilyen rádiótechnikai eszköz a jelforrás. Segítségével építhető olyan számítógépes bővítőkártya, amelyben a fizikai folyamatot erősítik, mintavételezik, és amely a kapott digitális jeleket átadja egy számítógépnek. A generált sorozatot folyamatosan ellenőrizni kell, mert a fizikai folyamatot befolyásolja a környezete, amely enyhébb esetben az eloszlás megváltozását, súlyosabb esetben degenerációt okoz.

A következőkben leírjuk egy PC-be illeszthető (vagy azzal más módon – pl. USB – összekapcsolható) fizikai generátorkártya felépítését. Ennek a fizikai véletlenszám generátornak az alapja, a zajforrás egy úgynevezett Zener dióda. Az ezen átfolyó áram feszültsége alapvető fizikai törvényszerűségek miatt kismértékben ugyan, de véletlenszerűen ingadozik. Az ingadozás „frekvenciája" 0 és néhány száz (200) kHz között változik (1. ábra).



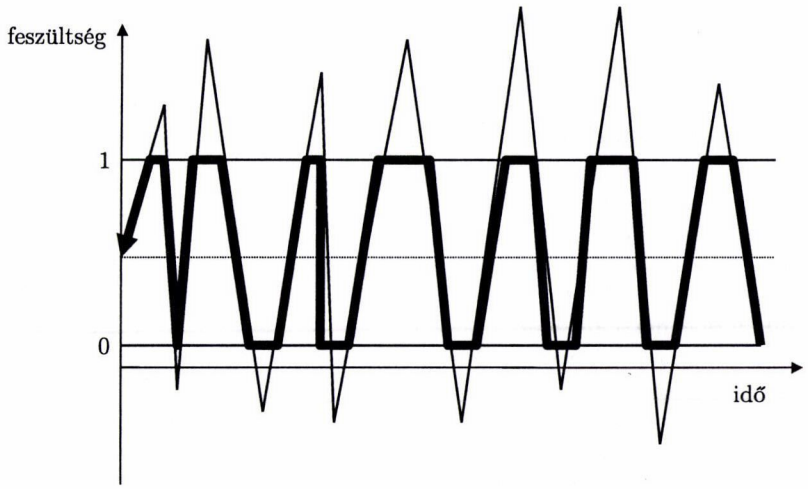
1. ábra

A kártyára épített elektronika ezt az ingadozást több fokozatban erősíti, majd a jel „közepén" szimmetrikusan kijelöl egy intervallumot, amelynek két szélső értéke lesz a 0 és az 1 érték (2. ábra).

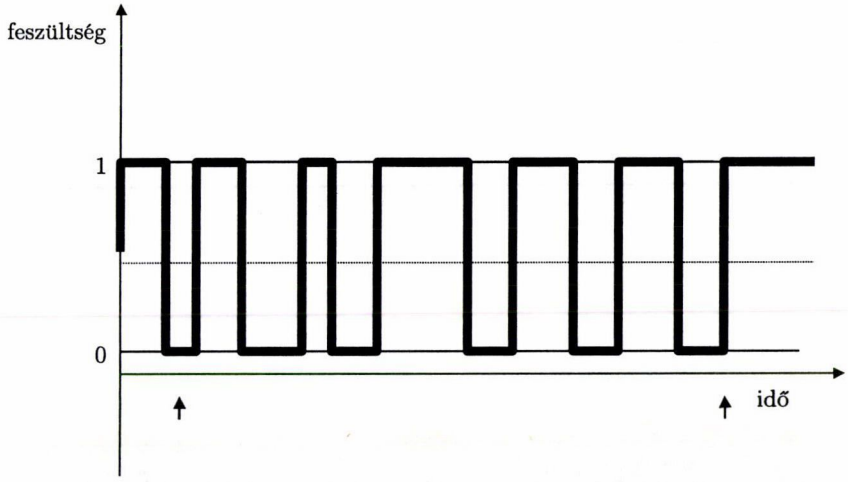
A jelet ezután négyszögesítjük. Ezután kerül sor a mintavételezésre, amelynek frekvenciája néhány ezer bit másodpercenként, vagyis a mintavételezés frekvenciája nagyságrendekkel kisebb a kiinduló jel frekvenciájánál, ami a szomszédos minták függetlenségét biztosítja (3. ábra).

A generátorkártyán két független jelforrás és mintavevő rendszer működik, a két forrásból származó biteket modulo 2 (xor) összeadva kapjuk azt a bitet, amely a generátor outputja lesz.

A PC-be illesztett kártyához egy, a PC operációs rendszerén futó driver tartozik, amely statisztikai ellenőrzéseket is végez az outputra, s szélsőséges esetben



2. ábra



3. ábra

riaszt. Szükséges még egy applikáció, amely segíti a kártya installálás utáni kalibrálását, és statisztikai teszteket végez, amelyek eredményét a felhasználó számára meg is jeleníti. Többszörösen hangsúlyozzuk **statisztikai tesztek folyamatos elvégzésének** szükségességét.

A legjobb beállítás mellett is eltér kissé a generált 0 és 1 bitek gyakorisága. Ezen ritkítással szokás javítani. Ennek során a sorozatot párokba osztják, és az azonos biteket tartalmazó párokat elvetik. Ha a pár bitjei különböznek, akkor az első bitet megtartják, a másodikat szintén elvetik. Jelölje p a ritkítás előtti sorozatban a 0 bit gyakoriságát, és tegyük fel, hogy a fizikai generátor bitjei független soroza-

tot képeznek. Ekkor annak a valószínűsége, hogy egy bitpár első bitjét megtartjuk a ritkítás után $= 2p \cdot (1-p)$. Így páronként a megtartott bitek számának várható értéke is $2p \cdot (1-p)$. Egy bitpár mindkét bitjét $p^2 + (1-p)^2 = 1 - \big(2p(1-p)\big)$ vetik el. Egy bitpár elvetése geometriai eloszlású ezzel a paraméterrel. Nyertük a következő tételt:

TÉTEL. *A fenti meghatározás és a függetlenség feltevése mellett az elvetett bitpárok számának várható értéke n bitpár esetén:*

$$\boxed{n\big[p^2 + (1-p)^2\big].}$$

Ez a várható érték szemléletesebb alakot nyer, ha p helyett $p = 1/2 + d$ összefüggést használjuk, ahol a d előjeles szám a várt 1/2-től való eltérést méri, tehát egy kis mennyiség.

$$p^2 + (1-p)^2 = \frac{1}{4} + 2\frac{1}{2} \cdot d + d^2 + \frac{1}{4} - 2 \cdot \frac{1}{2} \cdot d + d^2 = \frac{1}{2} + 2 \cdot d^2.$$

2.2. Pszeudo véletlenszámok generálása

Mint már említettük, az atombomba gyártásánál felmerült nagyon nagy mennyiségű véletlenszám előállításának az igénye. Fából vaskarika kívánalomként az is felmerült, hogy ezek reprodukálhatók legyenek. Neumann János 1944-ben javasolta azt, hogy erre a célra álvéletlen számokat használjanak. Valódi véletlenszámok helyett determinisztikus számsorozatot állítsanak elő, amelyek *statisztikailag ugyanúgy viselkednek, mintha valódi véletlen sorozatok lennének.* Ezeket a számsorozatokat ismételten elő lehet állítani, reprodukálni, ami a számítógépek hőskorában rendkívül fontos követelmény volt. Amint azt már mondtuk, az algoritmikusan előállítható, véletlenszerűen viselkedő determinisztikus számsorozatokat **pszeudo-véletlen** *(álvéletlen)* sorozatoknak nevezik.

A Neumann János által javasolt eljárás nagyon egyszerű volt. Kiindulásként választott egy n-jegyű (páros) számot kezdőértéknek. A pszeudo-véletlen sorozat következő számának előállításához az utolsónak generált n jegyű számot négyzetre emelte, majd az így nyert négyzet középső n jegyéből álló szám lett a sorozat új száma. (Az n jegy szükség esetén nullákkal kezdődhet.)

A négyzetközép módszernek ma már csak történelmi jelentősége van. Pszeudo véletlen számsorozatok előállítására a leggyakrabban a *lineáris kongruencia generátorokat* használják. Ezt a módszert D. A. Lehmer javasolta 1949-ben, lásd Lehmer (1951). Ehhez választanak egy $R(0)$ **egész kezdőértéket, egy** A **multiplikatív és egy** B **additív konstanst**, valamint egy nagy M **modulust**. Ezek segítségével az utolsónak előállított $R(N)$ számból kiszámítják az $R(N+1) = A \cdot R(N) + B \pmod{M}$ egész számot. Ebből egy további $U(N)$ álvéletlen valós számot M-mel való osztással kapunk. Ez az osztás egy bizonyos fajta standardizálást szolgál: az $U(N)$ számok mindig 0 és 1 közé esnek. A számítógépek RND generátora ilyen

számok sorozatát szolgáltatja. Innen a $\{0, 1, \ldots, k-1\}$ halmazba eső egyenletes eloszlású egész számokat az

$$X(N) = \lfloor k \cdot U(N) \rfloor$$

képlet ad meg (adja vissza).

A paraméterek megválasztásával és az előállított sorozat statisztikai vizsgálatával kapcsolatos kérdésekről D. Knuth (1987) 2. kötete ad kimerítő tájékoztatást.

Az algoritmikus eljárások gyors végrehajtására különböző célgépeket szerkesztettek, melyek valódi véletlen sorozatok helyett determinisztikus, de véletlenszerűen viselkedő sorozatokat alkalmaznak. Gyors hardver megoldást tesznek lehetővé a maximális periódusú Lineáris Visszacsatolt Shift Regiszterek (angol rövidítéssel LFSR). Nagy periódusú regiszterek megkeresése, elemzése a *Galois testek elméletében* való elmélyülést igényel. Maximális periódusú regiszterek kereséséhez kulcsreferencia: Golomb (1967), magyar témaismertetést ad meg Nemetz–Katona (1969).

2.3. Hibrid generátorok

Ahogy azt a 2. fejezet elején említettük, a véletlenszámok tömeges előállításának hőskorában megszületett az az ötlet, hogy bármilyen nagy munkával is, de célszerű *egyszer* nagymennyiségű véletlenszámot előállítani, és ezeket közkinccsé téve belőlük bárki, bármikor tetszőlegesen sokat fel tudjon használni. Természetesen ez a megoldás nem egy megbízható megoldás a kriptográfusok számára. Hiszen éppen a legfontosabb követelmény, a kiismerhetetlenség sérül meg. Ezt kell orvosolni ahhoz, hogy ilyen típusú megoldás számukra is használhatóvá váljon. Ezt több lehetőség is segítheti:

- A CD-ROM tárolók megjelenése az általában egyetlen alkalmazáshoz szükséges véletlenszám mennyiség sokszorosának megőrzését teszi lehetővé.
- A CD-ROM-okat abszolút biztonságos körülmények között lehet teleírni valódi véletlen (fizikai) generátorok segítségével, miközben a teleírás történhet szubjektív (személyi) beavatkozás nélkül.
- Véletlenszerű kezdettel kis blokkokat úgy lehet kiemelni a CD-ROM-ról, hogy még a kezelő sem képes felismerni, honnan származnak, tehát védelmet lehet biztosítani a saját kezelő személyzettel szemben.
- A kivételi program speciális, egyedi paramétereket tartalmazhat, testre szabható. Rövid, valódi véletlen kezdőértékeket használhat, melyek manuálisan is előállíthatók (például klaviatúra segítségével, egérmozgatásos eljárással).
- Nincs szükség gyors generálásra, tehát a generálást ellenőrző statisztikai tesztek halmaza a szokásosnál nagyobb lehet.
- Rendkívül gyors felhasználást tesz lehetővé.
- Hírközléses pár-kapcsolat esetén „egyszeri átkulcsolásra" is alkalmazható.

A hibrid generátorokról egy részletesebb analízis érhető el Nemetz–Papp (1998) cikkében. A szerzők CD-RANDOM generálási módszerrel meg is valósították a fenti elképzeléseket.

3. Véletlen kulcsgenerálás

3.1. Kriptográfiai háttér-minimum

Napjainkban a kriptográfia két elkülönült világa él egymás mellett:

- Hagyományos, *egyetlen* titkos *kulccsal* működő rejtjelzés, a **titkos kulcsú rejtjelzés**.
- *Két kulccsal* működő, **nyilvános kulcsú rejtjelzés**

Közös bennük, hogy mindkettő a leendő felhasználásnak megfelelően választott egyértelműen dekódolható *kódok nagy halmazát* választja ki. Egy továbbítandó (tárolandó) *nyílt üzenet*et mindig ezek egyikével kódolnak (rejtjeleznek). A lehetséges kódok halmazára mint kódhalmazra hivatkoznak, és elemeit valamilyen számszerű paraméterrel (sorszámmal vagy vektorral) azonosítják. Ezt az azonosítót a kriptográfiában kulcsnak nevezik. Szokásos azonban az is, hogy kulcsról csak egy titkosítandó szöveg esetén beszélnek. Egy **üzenet kulcsának** *nevezik annak a paraméternek az értékét, amellyel az adott üzenetet titkosítják.*

Mivel a kódhalmazt a támadó előtt ismertnek kell feltételezni, így az egyetlen bizonytalansági faktort a kulcs megválasztása jelenti. Ezt tehát a legnagyobb bizonytalanság biztosítása mellett, véletlenszerűen kell kiválasztani. Számunkra ebben tér el leginkább egymástól a két kriptográfiai rendszer.

A két rendszer különbözőségének igazi oka a lényegesen eltérő felhasználás.

3.1.1. Titkos kulcsú kriptográfia. A klasszikus kriptográfia módszereit elsősorban a **zárt hálózatok használják.** Ilyenek a katonai hírközlés, a diplomácia területén fordulnak elő. Fő jellemzőjük, hogy új felhasználó ilyen rendszerbe nem tud önkényesen, saját akaratából belépni. A kulcsokat többnyire egy központ generálja és osztja ki. A kapcsolatba lépő párok előre ismerik a felhasználható kódokat (transzformációkat), előre meg tudnak állapodni az egymás után alkalmazandó kulcsokban, vagy azok megválasztására alkalmazott algoritmusokban. Ezeket egymás között abszolút biztonságosan ki tudják cserélni, és elvben gondoskodhatnak abszolút biztonságos őrzésükről is. A titkos kulcsú kriptográfiában ennek megfelelően a kulcs megválasztásához *egyetlen véletlenszám* generálására van szükség, bár ez a szám akár egy végtelen számsor is lehet. Röviden felidézzük a ma leggyakrabban használt titkos kulcsú rejtjelző algoritmusokat.

DES: 1976 decemberében a National Bureau of Standards, USA, bejelentett egy új „Nemzeti Adattitkosítási Szabványt" (FIPS No. 46), amelyben a Data Encryption Standard, DES, **rejtjelző algoritmust** szabványosították. Leírása megtalálható az USA szabványok gyűjteményében. A DES 64 bites (8 byte-os) nyílt üzenetel et képez le ugyancsak 64-bites rejtjeles üzenetekbe 56 bit nagyságú kulcsméret mellett. Az IBM által **kifejlesztett blokk algoritmus** *Shannon keverő transzformációját* használva biztosítja, hogy a blokkon belül a kimenet minden bitje függ a bemenet minden bitjétől. A szabvány első változata tartalmazta azt a kikötést is, hogy csak hardverrel implementált változata használható, és az USA kormányzat megtiltotta, hogy ezt a hardver kivitelezést exportálják. Magát az algoritmust

5 évenként biztonsági vizsgálatnak vetették alá. Ez utoljára 1994-ben történt meg, amikor 1998-at jelölték meg a felhasználhatóság utolsó határának.

A DES jelenlegi legfejlettebb változata a „Triple Des, 3-DES". Ez vagy kettő, vagy három 56 bites kulccsal dolgozik. Az üzenetet először az első kulccsal rejtjelzik normál DES módban, majd a második kulccsal a megoldó algoritmust alkalmazzák. Az így nyert közbülső szövegre alkalmazzák ismét az első, három kulcsos rendszerben a harmadik kulcsot.

A DES megfejthetőségére utaló publikációk sorát Hellman egy 1977-ben tartott előadása nyitotta meg, aki a teljes kipróbálást is kivitelezhetőnek tartotta megfelelően épített hardver segítségével. A DES halálához a döntő csapást Biham és Shamir munkássága adta meg, akik egy újonnan kifejlesztett módszer, a „Differential Cryptanalysis" segítségével adtak meg egy fejtési eljárást. 1994-ben Matsui egy más típusú, ún. lineáris kriptanalízis módszert adott meg, amely már gyakorlatilag kivitelezett fejtésről számol be.

A véletlenszám generálás számára a lényeg: **egy üzenet rejtjelzéséhez 56 bit valódi véletlen bit szükséges.**

AES: A DES haldoklása láttán az amerikai *National Institute of Standards and Technology (NIST)* pályázatot hirdetett egy új amerikai (és világ) rejtjelzési algoritmusra, amely **Advanced Encryption Standard (AES)** néven volt hivatott átvenni a korábbi uralkodó DES szerepét.

Az NIST kidolgozott egy olyan követelményrendszert, amelyet a pályázatoknak ki kell elégíteni. A követelményrendszer blokk-rejtjelző algoritmust ír elő *128 bites blokkmérettel és 128-256 bites kulcsmérettel.*

A NIST *1999. március 22–23. között Rómában* egy szakértői konferenciát rendezett a beérkezett „legjobb" 15 pályamű értékelésére. Itt a legjobbnak ítélt öt algoritmust választották ki. 2001 szeptemberében hirdették ki a győztest, amely a Rijndael algoritmus lett. Részletesebben: Nemetz–Papp (2001).

A véletlenszám generálás számára a lényeg: **egy üzenet rejtjelzéséhez maximálisan 256 bit valódi véletlen bit szükséges.**

Véletlen átkulcsolás (one time pad). Ez az eljárás bizonyítottan, elméletileg fejthetetlen rejtjelezési eljárás. Ennek során a kulcssorozat minden elemét egymástól függetlenül választják ki egyenletes eloszlással abból az ábécéből, amelynek a jeleit a dokumentum is használja. A sorozat ugyanolyan hosszú, mint a dokumentum. A rejtjelzés abból áll, hogy egy adott összeadó tábla szerint a dokumentum és a kulcssorozat egymás utáni jegyeit összeadják. Mivel a partnerek ismerik egymást, képesek biztonságos csatornán a kulcssorozatot előre egymással kicserélni.

A véletlenszám generálás számára a lényeg: **egy üzenet rejtjelzéséhez ugyanannyi valódi véletlen betű szükséges, ahány betűből a rejtjelezendő szöveg áll.**

3.1.2. Nyilvános kulcsú kriptográfia. Nyilvános hálózatok: Az információs társadalomban egymást nem ismerő partnereknek kell megbízható elektronikus kapcsolatot teremtve titkos üzeneteket váltani. Erre a lehetőséget a nyilvános kulcsú kriptográfia adja meg.

Felhasználói két egymással kapcsolatban álló kulcsot választanak maguknak. Az egyik kulcsot nyilvánosságra hozzák, a másikat szigorúan titokban tartják. Az egyik kulcsot a feladó használja a rejtjelezésre, a másik kulcsot a címzett a megfejtésre. Közös rejtjelzési algoritmust használnak, amelyben könnyű kódolni, és a titkos kulcs birtokában a dekódolás is gyors. Az illetéktelen dekódolás azonban gyakorlatilag nem kivitelezhető. Ha a rendszer egy résztvevője egy másik résztvevőnek akar titkos üzenetet küldeni, akkor kikeresi a nyilvántartásból a címzett nyilvános kulcsát, és az ismert kódolási rendszerben ennek a paraméternek megfelelő kulccsal rejtjelzi az üzenetet. Ezt az üzenetet csak a hozzátartozó kulccsal, a címzett titkos kulcsával lehet visszaállítani (dekódolni), tehát ezt csak az illetékes címzett tudja elolvasni.

Megismételjük, hogy a nyilvános kulcsú rendszerek olyan közös rejtjelzési algoritmust használnak, amelyben a rejtjelzést a nyilvános kulcs birtokában könnyű elvégezni, de pusztán ezzel a kulccsal a dekódolás gyakorlatilag nem kivitelezhető. A titkos kulcs segítségével azonban a dekódolás is gyors művelet. A transzformáció végrehajtása gyors, az inverzének a meghatározása azonban rendkívül bonyolult. Az ilyen transzformációkat egyirányú transzformációnak nevezik. Ilyenek keresése a matematika nehéz feladata.

Az RSA algoritmus a modulo aritmetikában az ismeretlent hatványban tartalmazó egyenletek megoldásának nagyfokú bonyolultságát használja ki, így megfelelő nagyságú modulus esetén a megoldás technikai kivitelezhetetlensége szolgáltatja a biztonságot. A „megfelelő nagyság" itt kritikus szerepet játszik: a kezdetben biztonságosnak ítélt 512 bit hosszú kulcsok helyett ma a gyakorlatban 1024-nél rövidebb kulcsot (modulust) csak elvétve használnak.

Az RSA első alkalmazásai között szerepel a hozzáférés-védelem, digitális aláírás és az üzenethitelesítés. A témába jó bevezetést ad Akl (1983), egy összefoglaló helyzetképet pedig Simmons (1988).

A véletlenszám generálás számára a lényeg: **egy üzenet rejtjelzéséhez**

- **két nagy véletlen prímszám,**
- **egy nagy véletlen exponens szükséges.**

Az, hogy „nagy", azt jelenti, hogy a két prímszám szorzatának nagyságrendje 1024 bit, és általában ugyanekkora az E és D exponensek nagyságrendje is, de legalább az egyiké.

3.2. Véletlen kulcsgenerálás a klasszikus kriptográfiában

A klasszikus kriptográfia rövid ismertetéséből látszik, hogy két rendkívül eltérő nagyságú kulcsmérettel állunk szemben. Az egyik típus véges automatákkal dolgozik, ε nyílt szöveg rövid blokkjain hajt végre transzformációkat. Ennek megfelelően a kulcsai viszonylag rövid bitsorozatok. Előállításuk központilag, szakemberek irányítása mellett történik.

A másik eset a végtelen, véletlen átkulcsolás esete. Ehhez folyamatosan működő fizikai generátorokat használnak. Az így generált sorozatokat valahogyan tárolni és továbbítani kell. Ennek egyik módja a CD-ROM-on, DVD-ROM-on történő tárolás

és továbbítás. A generálás módját fentebb elfogadható részletességgel ismertettük, így most áttérünk a nyilvános kulcsú rendszerekre.

3.3. Nyilvános kulcsok generálása

A véletlen kulcsgenerálás egyik feladata a véletlen exponens megválasztása. Ez egy egyszerű feladat: adott nagyságrendű véletlenszámot kell pusztán generálni. Ennek a nagyságrendjét számítástechnikai, forgalmi, tehát nem matematikai meggondolások határozzák meg.

Az izgalmas kérdés a véletlen prímszámok megválasztása. Az általánosan elfogadott gyakorlat szerint a megadott nagyságrendben választanak egy véletlenszámot, majd algoritmikusan megkeresik a legkisebb, nála nagyobb prímszámot.

Az RSA titkos és nyilvános kulcspárában szereplő modulus ma javasolt minimális mérete 1024 bit, de mesterkulcsokhoz, hosszabb idejű védelem esetén 2048 bitet javasol az RSA cég. A generálás véletlenszámok alapján folyik. A kulcspárhoz két prímet kell generálni. A generálás legtöbbször úgy történik, hogy a véletlen input alapján az algoritmus egy-egy 512 bites páratlan számból indul. A véletlenszámot kettesével növelik, és a kapott számokat egy vizsgálatnak vetik alá, hogy az úgynevezett „pszeudo-prím" szám-e (potenciális prím-e)? Ez egy nem determinisztikus prímteszt. Az ellenőrzésre a Rabin-Miller tesztet használjuk, hússzoros iterációval. Korábban, amikor még 512 bites modulust is elfogadhatónak tartottak, a két véletlenszám nagyságrendjére különböző előírásokat ellenőriztek, hogy azok elég messze vannak-e egymástól. A következő gondolatsor mutatja, hogy erre 512 bites véletlenszámok esetén nincs szükség.

LEMMA. *Egy X valószínűségi változó akkor és csak akkor egyenletes a $\{0, 1, \ldots, 2^r - 1\}$ halmazon, ha X bináris kifejtésében minden bit egymástól teljesen függetlenül egyenlő $(1/2)$ valószínűséggel veszi fel a 0 és 1 értéket.*

LEMMA. *Legyen $x_i, y_i \in \{0, 1\}$*

$$X = x_1 \cdot 2^{r-1} + x_2 \cdot 2^{r-2} + \ldots + x_{r-1} \cdot 2^1 + x_r$$

és

$$Y = y_1 \cdot 2^{r-1} + y_2 \cdot 2^{r-2} + \ldots + y_{r-1} \cdot 2^1 + y_r$$

két egyenletes eloszlású valószínűségi változó a $\{0, 1, \ldots, 2^r - 1\}$ halmazon. Legyen továbbá d az a legkisebb index, amelyre $x_d \neq y_d$. Akkor $2^d \leq |X - Y| < 2^{d+1}$.

LEMMA. *Legyen X és Y két egyenletes eloszlású valószínűségi változó a $\{0, 1, \ldots, 2^r - 1\}$ halmazon. Ekkor eltérésük várható értékére fennáll, hogy*

$$\frac{1}{2} \cdot 2^{r-1} < E\{|X - Y|\}.$$

Ez $r = 512$ esetén nagy eltérést jelent: $1/2 \cdot 2^{r-1} \approx 10^{153}$, tehát a várható különbség külön „rendszabály" nélkül is nagy.

3.3.1. Billentyűs véletlenszám generátor. A generálás alapötlete az, hogy a billentyűzet gombjainak egymás utáni leütései között eltelt idő felhasználásával egy hexadecimális sorozatot állítunk elő. A billentyű-leütések közti idő kihasználása nem új ötlet. Létezik a PGP programoknak olyan változata, amely ezt a módszert használja. Ezek a programok azt kérik a felhasználótól, hogy üssön le (önkényesen választott) billentyűket, mondjuk ötvenszer.

Ami mégis újjá teszi módszerünket, az az, hogy a leütendő billentyűk sorozatában nem hagyunk helyet az önkényeskedésnek. Ezt a sorozatot a PC beépített pszeudo véletlenszám generátorának véletlenszerű indításával a leütendő sorozatot kiíratjuk a képernyőre, és a kliensnek ezt a sorozatot kell reprodukálnia. Hibás leütést a gép nem fogad el. Az eltelt időt (milisec) 16-tal osztva a maradék lesz az aktuálisan generált véletlen hexadecimális jegy.

Az ellenőrzéshez készítettünk egy C++ programot, és a vele generált véletlen outputot statisztikailag teszteltük. A tesztek megnyugtató eredményre vezettek. A program lépései:

Algoritmus

1. **Lépés:** Megadunk egy nyomtatható (képernyőre írható) karakterekből álló ABC$ stringet (vagy mátrixot). Legyen a stringben (mátrixban) szereplő jelek száma M, ami a program paramétere. A stringben szereplő karakterek ismeretén kívül bizonytalansági faktort jelenthet a benne szereplő jelek sorrendje is.
2. **Lépés:** A program közli, hogy a generált véletlen számokat mindig az [rndout.doc] file-ba menti. Felszólítja a felhasználót, hogy az exe file futtatása előtt gondoskodjon az esetleg ilyen nevű file tartalmának a mentéséről, mert ezt a program felülírja.
3. **Lépés:** A program rákérdez a generálandó hexadecimális jegyek N számára. Ez a szám 1 és 500 közti tetszőleges egész lehet.
4. **Lépés:** Állítsuk be a PC pszeudo-véletlen generátorát egy véletlen kezdőértékre, és generáljunk egymás után N pszeudo-véletlen számot az $1, 2, \ldots, M$ egész számok közül.
5. **Lépés:** Minden egyes generált pszeudo-véletlen egész esetén írjuk ki a képernyőre az ABC$ string ennyiedik betűjét. (Ez szervezhető $N = 280$ esetén például 4 darab 70 betűs sorba). Ezzel kezdődik a véletlen-hexa generálás.
6. **Lépés:** Üssük le a billentyűzeten egymás után a képernyőn levő N karaktert, miközben a PC méri és kódolja az egymás utáni leütések közti időt, és leírja (feljegyzi, tárolja) a generált értéket. A kiírás megint szervezhető 4×70-es mátrixba, ahogy ezt a konkrét példánk teszi is.

Algoritmus vége.

Teljesen lényegtelen, hogy a számítógép milyen pszeudo-véletlenszám generátort alkalmaz. A program működésének bemutatására $N = 4 \times 70$ paraméter mellett megadunk egy **leütendő sorozat**ot, majd a kiírt sorozat leütésekor létrejövő véletlenszám sorozatot. Megadunk egy példát:

$N = 280$. A leütendő sorozat:

```
3gkmh%g7wl   g%zi3$6e=,   wl3jz-bpfm   t6ibcbp8nt   2=a,%p4=ma
24hrcrcp08   ym7yms5x59   ecx4d-tpzn   3$@h5tpa60   37pfakprwu
a34-yx@0pv   0jh68$#e7!   bbizueg,kw   cynu.%r6$8   mesw%wxubn
w6b3phq9pz   d!a$%ny,x0   kwkzl!%=rf   03ap4xz9%3   za4yrct3o@
xmg0bu@d!1   su=a%cyv!2   .,9uton6yx   oo9@x$zcph   gmazktp3q3
t1zhw%!t-j   ok,t$dp8,v   r4wzg8%,xo
```

A generált sorozat:

8 2E5F 34 3C5	6687D4A 1574	F 2D60BE3 2 1	5B 1 0 5D9595	70 55 1 0 49C7
65D2EF56BB	9D 1A8E27A7	7 264AA 6BC 8	5E906AD0 0C	C6BF05 30 66
70 12CC70 2A	C0B 1 1 8E67 1	A35D4EDED6	C0EB221A4C	9BBBE0 043F
1 3 27 2E40B 2	878 1BCCBC9	AF 3 2 94E75 D	3 2 0 86A97B 2	1 45 276 8 6 0C
65EB2B9 1 6 1	9 6CE0 7F 28 2	4 2 7 8 8 2DB7 6	DBC9E59E1 1	DCB0956BF8
CBF5559E5C	37CFB0CE5D	6ABB 36 6 8E 0		

A generált sorozat gyakoriság-eloszlása:

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
22	17	22	11	12	24	23	19	14	15	12	25	21	14	19	10

A 15 szabadsági fokú χ^2-statisztika értéke: 21.485, tehát a sorozat megbízhatóan véletlen.

Hat további 4×70-es sorozat: gyakoriság-eloszlása rendre:

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
12	16	16	20	17	15	19	26	22	17	25	15	16	14	18	12
20	13	18	22	13	16	16	21	10	14	23	22	19	13	25	15
10	19	19	22	22	15	13	20	17	22	18	16	23	17	12	15
15	12	17	21	14	11	15	23	24	22	15	12	16	23	16	24
16	20	18	17	17	16	19	17	19	18	11	12	23	17	21	19
22	18	24	17	19	13	17	21	15	17	19	14	23	17	10	14

A hat 15 szabadsági fokú χ^2 **próba-statisztika értékei:**

Minta száma	1	2	3	4	5	6
Khi-értékek	14,28	16,46	12,80	18,06	7,66	12,46
Megbízhatóság	0,49	0,31	0,62	0,16	0,89	0,58

Ezen adatok alapján nem lehet megcáfolni a véletlenszerűség hipotézisét.

Bemutatunk egy másik, nagyon egyszerű tesztet is.

2. teszt: A statisztika a 4×70-es mátrix azon oszlopainak a számával egyenlő, amelyekben mind a 4 hexa-szám különböző, tehát oszloponként bináris változó:

$$P(\text{mind különböző}) = \frac{(15 \cdot 14 \cdot 13)}{(16 \cdot 16 \cdot 16)} = 0{,}6665,$$

$$P(\text{nem mind különböző}) = 0{,}3335.$$

A minta gyakoriságai: mind különböző: 46,
nem mind különböző: 24.

Gyakoriság: 0,6286, illetve 0,3714; az adódó χ^2 próba értéke: 0,453038. Az 1 szabadsági fokú kritikus értékek:

P	0,1	0,05	0,01
χ^2 érték	2,71	3,84	6,63

Tehát legalább 0,99 annak a valószínűsége, hogy a 70 elemű véletlen minta ekkora vagy ennél nagyobb χ^2 értéket eredményez.

Mindkét esetben a leggyakrabban használt statisztikai próbát, a χ^2 **próbát** alkalmaztuk.

3.3.2. Internetes véletlenszám generátor. Kriptográfiában olyan véletlenszámokra van szükség, amelyek kiismerhetetlenek, az előzményekből nem jósolhatók meg. Ezért javasoljuk olyan interneten elérhető fájlok használatát, amelyek azonosítását az ellenfél várhatóan nem tudja elvégezni. A módszer alkalmazhatóságát elméletileg a stacionárius, ergodikus sztochasztikus folyamatok egyik határeloszlástétele garantálja.

A program inputja egy, az internetről letöltött file. A program egy <input.txt> nevű file-t vár bemenetként, tehát a levett file-t erre a névre át kell nevezni. Ha nem talál ilyen nevű file-t a program, akkor hibát jelez.

A generált véletlen outputot egy <randout.txt> file-ba írja ki. Ha ilyen file létezett, azt felülírja, ha nem létezett, akkor újat hoz létre.

Az algoritmus vázlatos menete a következő:

1. lépés: Letöltünk egy szövegfile-t, például újságcikket az internetről. Ezt a továbbiakban egy **karakter sorozatnak (byte sorozatnak) tekintjük.**

2. lépés: A letöltött sorozatból blokkokat képezünk, felváltva a hosszú aktív, illetve b hosszú passzív blokkokat. Tekintsük a-t és b-t az algoritmus paramétereinek, amelyek ideális értékét statisztikai tesztek eredményeinek felhasználásával lehet meghatározni.

3. lépés: Tekintsük az aktív blokkok elemeit egész számoknak. Ezt megtehetjük úgy, hogy a byte-okat a bináris alakjuknak megfelelő számoknak tekintjük, de paraméterként egy egyszerű helyettesítést jelentő táblázattal is átértékelhetjük őket. A véletlenszámok generálása során a helyettesítő tábla semmilyen befolyásoló szerepet nem játszik, bármikor tetszőlegesen megváltoztatható. Megváltoztathatósága növeli viszont a bizonytalanságot. „Nagyjából" véletlenszerűen, automatikusan változtatva, a generálás konkrét megvalósítását is elrejthetjük a kezelőszemélyzet elől.

4. lépés: Választunk egy a hosszúságú $\mathbf{c}$ vektort az $\{1, 3, 5, 7, 11, 13\}$ halmaz elemeiből, lehetőleg valamilyen pszeudorandom generátor segítségével.

5. lépés: Az aktuális aktív blokk és a **c** vektor modulo 16 vett szorzata adja a következő véletlen számot.

6. lépés: A b hosszúságú passzív blokkot figyelmen kívül hagyva a következő aktív blokkal dolgozunk tovább.

Az algoritmus-tervezet megbízhatóságának kiértékeléséhez készítettünk egy C++ programot, amely mindkét számot paraméterként használja. A program rákérdez, hogy a felhasználó milyen hosszú blokkokkal kíván dolgozni. Különböző a és b paraméterek mellett számos interneten elérhető (U1–U10) cikket dolgoztunk fel, és a készített „véletlen" sorozatot statisztikai tesztek segítségével vizsgáltuk.

cikk	U1	U2	U3	U4	U5	U6	U7	U8	U9	U10
0	60	24	24	49	35	19	36	49	52	68
1	45	9	15	48	25	36	31	57	49	61
2	58	21	14	51	22	24	36	52	41	60
3	73	20	20	53	24	24	34	47	41	45
4	61	21	25	64	31	31	41	47	45	55
5	35	22	23	53	30	30	32	36	43	50
6	61	23	23	63	18	24	24	44	50	50
7	48	15	22	49	21	31	40	49	43	40
8	63	27	26	68	22	24	38	54	53	48
9	53	24	19	51	20	20	26	46	39	53
A	61	21	20	63	21	23	32	53	62	45
B	55	28	23	51	22	20	33	55	44	52
C	59	20	20	60	24	32	39	51	45	48
D	58	16	22	67	17	33	37	43	53	39
E	57	19	26	61	29	38	36	52	56	63
F	42	35	27	61	19	33	34	41	51	45
χ^2	22,96	23,65	9,464	12,842	16,295	19,828	9,543	9,691	12,577	19,966

1. táblázat: Generált véletlen számok gyakoriságeloszlásai.

Az első két cikk χ^2 értéke a $P = 0{,}1$-es elfogadási határ fölé csúszik, a többié megfelelően alacsony, így a véletlenszerűség feltételezése elfogadható.

A statisztikai tesztek a Knuth (1987) könyv tesztjeinek adaptációi voltak, $p = 0{,}05$ hibavalószínűség mellett, khi-négyzet kiértékeléssel. Nagyobb aktív-blokk hosszúság egyenletesebb eloszlást eredményezett. A passzív blokkok szerepe az, hogy a program outputjaként előálló sorozat elemei kellő mértékben függetlenek legyenek. Mindkét blokk hosszának növelésével az eredmények véletlenségét növelhetjük, ám egyúttal csökken a fileból kinyerhető sorozat hossza is. A statisztikai tesztek eredményeként a 10 hosszúságú „aktív" és 3 hosszúságú „tétlen" blokkok használatát ajánljuk.

Az 1. táblázat 10 cikkből nyert gyakoriság eloszlásokat mutatja az $a = 10$, $b = 3$ paraméterek esetén. Az utolsó sor a kapott „véletlen-hexa" sorozatnak az egyenletes eloszlástól való khi-négyzet eltérését adja meg.

A cikkek, illetve egyéb dokumentumok Internetről történő elmentésekor vigyázni kell arra, hogy lehetőleg csak természetes nyelvű szöveg kerüljön a program inputjára. A weboldalakon jelentős részt kitevő html utasítások, gyakran ismétlődő sablonok (például tartalomjegyzék) felhasználása nagyban rontja az eredmény véletlenszerűségét. Így a *mentés másként (text típus)* illetve a *kijelöli mindet* menüpontok használata helyett biztonságosabb, ha manuálisan jelöljük ki a kívánt részt és a vágólapon keresztül bármilyen egyszerű szövegszerkesztő segítségével mentjük el a program számára.

4. A véletlenszerűség statisztikai ellenőrzése

Az eddigiekben többször hangsúlyoztuk, hogy szükség van olyan tesztekre, amelyek segítségével eldönthető, hogy egy adott sorozat tekinthető-e véletlenszerűnek. Ez a feladat része egy általános statisztikai feladatnak. A kérdés az, hogy egy adott megfigyelés sorozat származhat-e egy adott valószínűség eloszlásból. Ezt a kérdést valamennyi statisztikai tankönyv tárgyalja, így nem lenne célszerű erre az általános kérdésre időt/teret vesztegetni. A továbbiakban cél-orientált tesztekkel foglalkozunk.

A véletlenszerűséget *vizsgáló kriptográfiai hátterű statisztikai próbák*nak egy gyűjteményét írja le a **National Institute of Standards and Technology (2001)**. Ezek az eljárások jól használhatóak arra, hogy egy *bináris sorozatnak* a véletlenszerűtől való eltérését kimutassák. A tesztelőnek azonban figyelembe kell vennie, hogy a véletlenszerűtől való eltérések lehetnek egy gyengén tervezett generátor következményei, de felléphetnek a tesztelt bináris szekvenciában lévő anomáliák miatt is (vagyis, bizonyos számú hiba fellépése valószínű a véletlen sorozatokban, amelyeket egy meghatározott generátor állít elő). A tesztelőn múlik, hogy a teszt eredményeket helyesen értelmezze.

4.1. A NIST által támasztott követelmények

Megadott bináris sorozatok véletlenszerűségére, megjósolhatatlanságára és a tesztelésre vonatkozó statisztikai tesztek összeállításánál három alapvető feltételt kell ellenőrizni. Ezek:

- **Egyenletesség**: Egy véletlen vagy pszeudo-véletlen bitsorozat előállításának bármely időpontján egy nulla vagy egy érték előfordulása egyformán valószínű, vagyis mindegyiknek a valószínűsége pontosan $1/2$. A nullák (vagy egyesek) száma várhatóan $n/2$, ahol n a sorozat hossza.
- **Skálázhatóság**: Egy sorozatra alkalmazható bármely próba alkalmazható egy véletlenszerűen kiragadott részsorozatra is. Ha a sorozat véletlen, akkor minden kiragadott részsorozatnak is véletlennek kell lennie. Így bármely kiragadott részsorozatnak meg kell felelnie minden véletlenszerűségre vonatkozó próbának. (Természetesen a vizsgált részsorozatok száma csak elenyésző kis töredéke lehet az összes lehetséges részsorozatnak: Kolmogorov bizonyította,

hogy ha a részsorozat választó algoritmusok száma legfeljebb $\frac{1}{2}e^{2n^2(1-\varepsilon)}$, akkor mindig létezik (n, ε) véletlen bináris sorozat, ahol az n-nél rövidebb részsorozatok egyenletesek legfeljebb ε eltéréssel (ld. Knuth (1981) 2. kötet, 174. o.))

- **Konzisztencia:** egy pszeudo véletlenszám generátor viselkedésének konzisztensnek kell lennie a kiinduló értékek mindegyikére. Nem kielégítő, ha egy PRNG-t csak egyetlen kiinduló értékből származó output alapján, vagy egyetlen másik paramétere alapján tesztelünk, vagy ha egy RNG-t egy olyan output alapján tesztelünk, amely egyetlen fizikai outputból lett származtatva.

A NIST által összeállított követelményrendszerben javasolt tesztekre 2001-ben a Hungard Kft egy RNDtests.exe nevű programcsomagot készített. A programcsomag ismertetésével rendkívül célratörően lehet magát a NIST javaslatot is ismertetni. Ezért az ismertetésnek ezt a módját választottuk.

4.2. Az RNGtests.exe program tartalma

A program indításakor a képernyőn megjelenik egy „startlap", amelyet a 4. ábra mutat.

A lap első kérdésként rákérdez az ellenőrizendő file nevére, majd a browserrel ezt megkeresi. Ugyancsak rákérdez az input formátumára.

A program input formátumai

A program háromféle input-formátumot tud fogadni. Ezek:

- Egy bájtban egy bit. (Bit formátumban előállított sorozatok. Ilyenek például a véletlenszerűen generált hexadecimális sorozatok.)
- Egy bájtban 8 bit. (Hagyományosan készített file, mesterségesen bit-értékekre korlátozva.)
- 0,1-et tartalmazó ASCII sorozat. (Ez a fajta file a kifejezetten erre a célra készített, véletlen bitsorozatot tartalmazó file.)

A futtatás elején meg kell jelölni, hogy ezek melyike az alkalmazni kívánt sorozat.

Bemeneti adat annak a file-nak a neve, amely az ellenőrizendő adatsort tartalmazza. Meg kell jelölni azokat a teszteket, amelyeket a felhasználó a megadott file-on el akar végezni. A Maurer-, Linear Complexity-, Random excursion teszteket fizikai generátorok bevizsgálásánál ajánlott elvégezni, „normál" napi munka számára kevésbé ajánljuk. Ezeket a teszteket a gép gyorsan hajtja végre, az eredményt a tudakozó lap jobb alsó részében írja ki.

A program minden tesztnél kijelzi, hogy mekkora a teszt megbízható működéséhez szükséges mintanagyság. A tesztek lefutása után a képernyőn megjelennek a tesztek eredményei, és egy összefoglalt ítélet arról, hogy a sorozat véletlenszerű-e.

A döntés minden teszt esetén $\alpha = 0{,}01$ megbízhatósági szinten történik. Ez azt jelenti, hogy várhatóan 100 sorozatból egy sorozatot utasít el a próba úgy, hogy a sorozat véletlen volt.

4.3. Tesztfajták

Frekvencia teszt. Az ellenőrizendő sorozatot adott hosszúságú konzekutív blokkokra osztjuk, ezeknek a blokkoknak a valószínűségeit az egyenletesség és függetlenség feltételezése mellett kiszámítjuk. Az adódó valószínűség-eloszlásra alkalmazzuk a khi-négyzet próbát. A teszt feladata, hogy kimutassa, ha a sorozatban túl sok nulla vagy egyes van.

Futamhossz teszt. k hosszú futamnak nevezünk egy pontosan k darab egymás után következő, azonos bitekből álló részsorozatot, amelynek mindkét végén az ellentétes érték található. A teszt célja, hogy a különböző hosszú futamok statisztikailag várható számát összehasonlítsa a sorozatban megfigyelt értékekkel. A futamhosszak túl nagy vagy túl alacsony száma arra utal, hogy a sorozatban túl sok az oszcilláció.

Spektrális Diszkrét Fourier Transzformáció teszt. A bitsorozatban esetlegesen fellelhető periodikus jelekre érzékeny. A sorozatban vizsgálat előtt -1-re cseréljük a nulla értékeket, s a módosított sorozatra elvégezzük a diszkrét Fourier transzformációt. A kapott változók sorozata reprezentálja az eredeti sorozat hosszabb-rövidebb periodikusan előforduló mintáit. Az ezután készülő statisztika a normális eloszlást használja referenciaként.

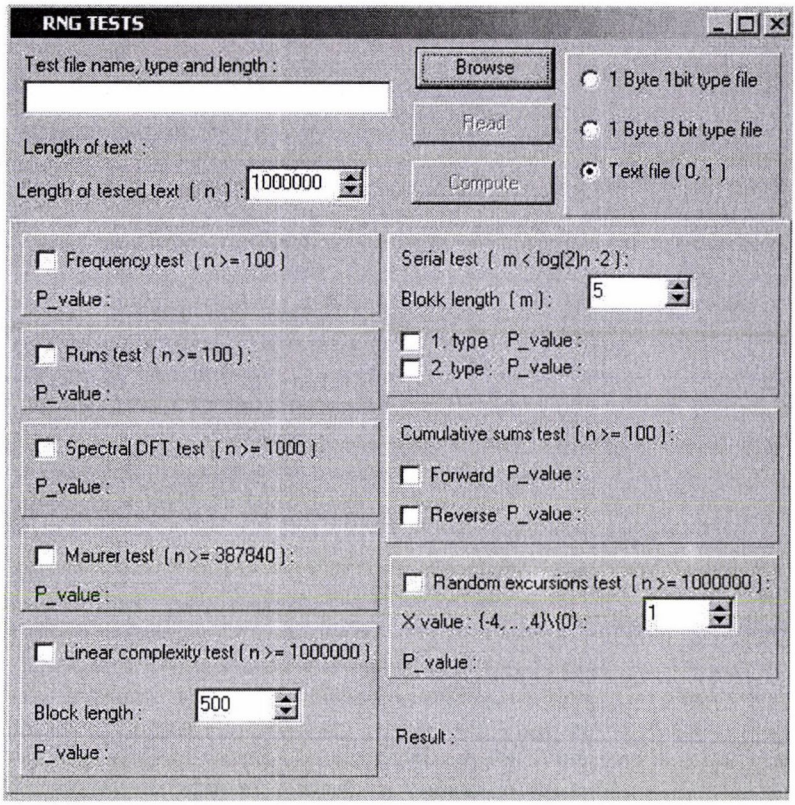
Maurer-féle univerzális entrópia teszt. A teszt fókusza az azonos minták (részsorozatok) közötti bitek számán van. A teszt azt méri, hogy a sorozat mennyire tömöríthető információvesztés nélkül. A szignifikánsan tömöríthető sorozatot *nemvéletlennek* tekintjük. A módszer alapja, hogy a sorozatot egy kezdeti inicializációs és egy maradék tesztrészre bontjuk. A teszt azt méri, hogy az inicializációs sorozat fix, k hosszú mintáival hogyan írható le a sorozat második, tesztszakasza. A statisztikához kapcsolódó referencia eloszlás a fél-normális eloszlás.

Lineáris komplexitás teszt. A teszt a lineáris shiftregiszterrel történő modellezhetőséget méri. A sorozatot k hosszú részblokkokra bontjuk, és minden blokkra végrehajtjuk a Berlekamp-Massey algoritmust, hogy meghatározzuk azt a legrövidebb lineáris shiftregisztert, amely az adott blokkot generálni képes. Véletlen esetben a regiszter hossza $k/2$ körüli érték. A kapott hosszakat minden részblokkra értékeljük.

Sorozat teszt. A teszt célja, hogy megvizsgálja az összes lehetséges 2^m darab egymást átfedő m-bites minta eloszlását a vizsgált sorozatban. Az adódó valószínűség-eloszlásra alkalmazzuk a khi-négyzet próbát. A teszt $m = 1$ esetben megegyezik a frekvencia teszttel. A teszt során két, részben különböző módon számított statisztikát is beállíthatunk.

Kumulatív összeg teszt. A 0/1 arány 0-tól való lokális eltérését méri. A teszt során a 0 értékeket -1-re módosítjuk és képezzük a részletösszegek s_i sorozatait, tehát az első i elemek összegeit. A statisztika az s_i értékek maximumát vizsgálva dönt a véletlenszerűségről. Beállítható, hogy a bejárást a sorozat melyik végéről kezdjük.

Véletlen bejárás teszt. Az előző tesztre épül. Megfigyeli, hogy az előző tesztben kiszámított s_i értékek hányszor egyeznek meg egy paraméterként beállítható



4. ábra. Az RNDtests.exe program tudakozó lapja

X értékkel. Minden X mellett független tesztnek tekinthető. A referenciaeloszlás a khi-négyzet eloszlás.

Hivatkozások

[1] Benson, R. L., Warner, M. (eds), Venona, Aegean Park Press (1996).

[2] Buszlenko, N. P.-Golenko, D. I., *Monte Carlo módszerek,* Műszaki Kiadó (1966)

[3] Friedman, *The Index of Coincidence and Its Applications in Cryptography,* Riverbank Publication, No. 22 (1920)

[4] Golomb, S. W., *Shift Register Sequences,* Holden-Day, San Francisco (1967)

[5] Jahnsson, B., *Random number generators,* Petterson, Stockholm(1966)

[6] Kahn, D. (1967), *The codebreakers,* MacMillan Co., New York

[7] Kerckhoffs, A., (1883) *La cryptographie militaire,* republished as „A Mathematical Theory of Cryptograhy", Bell System Techn. J. (1949)

[8] Knuth, D. E., *A számítógép-programozás művészete* 2., Szeminumerikus algoritmusok, Műszaki Kiadó, Budapest (1987)

[9] Lehmer, D. H., Linear Congruence Generators, in: *Proc.* 2nd *Symposium on Large-Scale Digital Calculating Machinery,* Harward Univ. Press, Cambridge (1951), 141–146

[10] Muha L. (szerk), Az informatikai biztonság kézikönyve, 10. Javított kiadás, Verlag Dashöfer, Budapest (2004)

[11] Nemetz, T. – Katona, G., Néhány megjegyzés a shift-regiszter generátorokról, *MTA Számítástechnikai Központ Közleményei,* No. 5 (június) (1969), 1–10.

[12] Nemetz, T. – Papp, P., Hybrid Random Byte Generators, in: *Global IT Security,* Papp, Gy.-Posch, R. (ed), Proc. of the 14th Int. Conf. on Info. Security, Wien (1998), 366–380

[13] Nemetz, T. – Papp, P., Belga Kriptográfusok Amerikában, *BYTE Magyarország,* V., No. 3. (2001), 38–40.

[14] Nemetz, T. – Wintsche, G., *Valószínűségszámítás és statisztika mindenkinek,* Polygon, Szeged (1999)

[15] NIST: Special Publication 800–22 (2001), *Statisztikai próbák a kriptográfiai alkalmazásoknál használt véletlen- és pszeudo-véletlen szám generátorokra* (javított kiadás: 2001. május 15.).

[16] Shannon, C. E., Communication theory of secrecy systems, *Bell Syst. Techn. J.,* **28** (1949), 656–715.

[17] Vincze, I., *Matematikai statisztika ipari alkalmazásokkal,* Műszaki Könyvkiadó, Budapest (1968).

[18] Yule, G. U. – Kendall, M. G., *Bevezetés a statisztika elméletébe,* Közgazdasági és Jogi Könyvkiadó, Budapest (1964).

GENERATING AND TESTING CRYPTOGRAPHICALLY SECURE RANDOM NUMBERS

Gergely Borbás, Tibor Nemetz, Pál Papp

In this paper we review some methods for generating real random numbers and pseudo-random numbers from a cryptographer's point of view. After summarizing the necessary crypto background, we examine how much random numbers are needed for different encryption algorithms.

We have developed two new methods for generating cryptographically secure pseudorandom numbers.Finally we deal with the methods known in the literature for testing the quality of a random number generator